Tipuri de date abstracte. Recursivitate

15 decembrie 2003

Programe compuse din mai multe siere

Implicit, obiectele declarate la nivel de sier sunt *unice* ^ntr-un program (doua declaratii ale aceluiasi identi cator ^n siere diferite reprezinta *același obiect*, v. curs 3).

⇒ obiectul va *definit* ^ntr-un singur sier, *declarat* ^n toate sierele ce-l utilizeaza. Declaratii care nu sunt de nitii:

{ pentru variabile: cu speci catorul `extern`

{ pentru functii, doar prototipul (antetul), nu si corpul functiei

Fazele compilarii:

{ compilarea ^n siere *obiect* `.c -> .o`

(cod masina, dar contine ^nca nume de variabile ^n loc de adrese xe)

{ editarea de legaturi (linkeditarea): referintele la un identi cator (*simbol*) din toate sierele obiect ^nlocuite prin aceeasi adresa

Obiectele cu speci catorul `static` nu sunt vizibile ^n afara sierului

⇒ acelasi identi cator poate refolosit pentru obiecte diferite

Structurarea programelor din mai multe siere

{ câte un sier pentru portiunile de cod care formeaza o entitate logica
{ cu un minim de interactiune (fara variabile globale nenecesare, etc.)
{ declaratiile de tipuri, functii si variabile ce trebuie exportate se pun ^ntr-un sier antet .h
{ acesta e inclus de ecare sier .c care ^l necesita
{ pentru a nu include/declara ^n duplicat, se poate ^ncadra ^n

```
#ifndef __FISIERULMEU_H
#define __FISIERULMEU_H
/* aici vine continutul propriu-zis */
#endif
```

Tipuri de date abstracte (cont.)

Decizii de proiectare:

{ ce operatii ca e incluse

{ daca se transmit obiecte sau doar pointeri la obiecte
(pointerii sunt necesari pentru functii care modi ca obiectul)

{ daca rezultatul unei operatii e returnat (eventual alocat dinamic),
sau depus ^ntr-un obiect speci cat (deja alocat) transmis ca parametru

{ daca functia returneaza un obiect, sau un cod de succes/eroare
(si obiectul e depus la adresa data de un pointer parametru)

Vezi exemple de cod pentru:

{ numere complexe, inclusiv suportul standard ^n C99 (`complex.h`)

{ matrici: variante de operatii cu 2 sau 3 argumente (incl. rezultatul)

{ multimi (de ^ntregi): cu numar x/precizat/variabil de elemente

Recursivitatea: Exemple

{ *în tipuri de date recursive*

Codul se scrie natural pornind de la de nitia recursiva a structurii:

Ex. *o listă este vidă sau un element urmat de o listă*

Se pot de ni atunci:

{ *membru*: e primul element, sau membru ^n coada listei

{ *șterge*: primul element, sau sterge din coada listei, etc.

La fel, se pot de ni recursiv functii care copiaza sau transforma liste.

{ *în analiza sintactică*

Productiile din gramatica unui limbaj sunt tipic recursive:

expresie ::= termen | expresie + termen | expresie - termen
termen ::= factor | termen * factor | termen / factor
factor ::= numar | (expresie)

Primele doua productii sunt *recursive la stânga*, pentru ca neterminalul din partea stânga a lui ::= apare si ca prim element ^ntr-o varianta

$\Rightarrow$ se pot transforma si implementa (vezi exemplu) folosind cicluri